Circuitos Digitais

Prof. Ricardo Pedroni
ricardopedroni@utfpr.edu.br
www.rpedroni.com.br

Máquinas de Estados

Prof. Ricardo Pedroni
ricardopedroni@utfpr.edu.br
www.rpedroni.com.br

Ementa

- Lembre!
- Circuitos Combinacionais x Sequenciais
- Otimização de Circuitos
- Projeto de Circuitos

Ementa

- Introdução
- Tipos de Máquinas
- Procedimento de Projeto
- Estilos de Codificação dos Estados
- O Problema de Sub/Sobre Especificação
- O Problema de "State Bypass"
- Máquinas Temporizadas

Máquinas de Estados

FSM

- Circuitos digitais podem ser classificados como:
  - Combinacionais: saída depende somente da entrada neste momento
  - Sequenciais: saída depende de estado anterior do sistema
- Máquinas de Estados (**FSM** → Finite State Machines) é uma terceira categoria?
  - **Não.** É uma técnica de projeto para circuitos sequenciais.

FSM

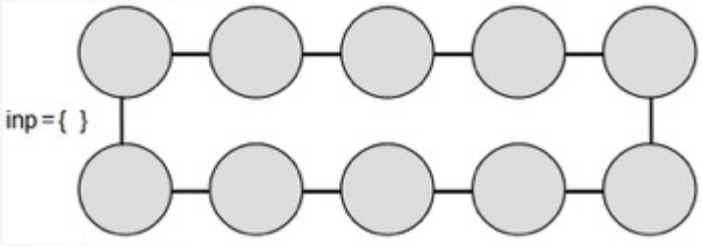
- Esta técnica é recomendada para qualquer circuito sequencial?
  - Somente para certos circuitos sequenciais
  - Sem muitos estados (seria inviável enumerá-los caso contrário)
  - Bom exemplo: Controladores (de elevador, de câmera, de CPU, …)
  - Mau exemplo: Contadores grandes (ex.: de 0 a 1000 → 1001 estados)

FSM

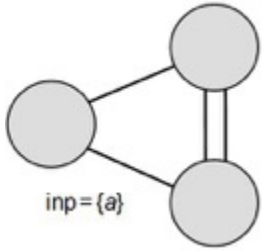
- O que é necessário para usar a técnica de FSMs?
  - Somente o **diagrama de transição de estados**
- O diagrama de transição de estados consiste de 3 elementos:
  - **Estados**: Lista de todos os estados possíveis do sistema
  - **Transições**: As condições de transição devem ser completas e complementares
  - **Saídas**: A lista deve ser a mesma em todos os estados (para hardware)

FSM

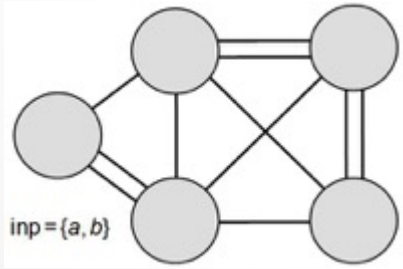
- Exemplos de Candidatos a técnica de FSMs:



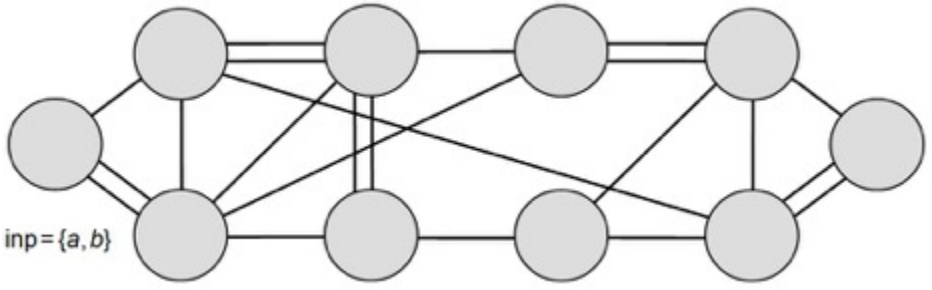
Bem fraco
(uma espécie de contador)



Fraco
(bem poucos estados, somente uma entrada)



Forte
(mais estados, mais interconexões, possivelmente mais do que uma entrada e envolvendo tempo)



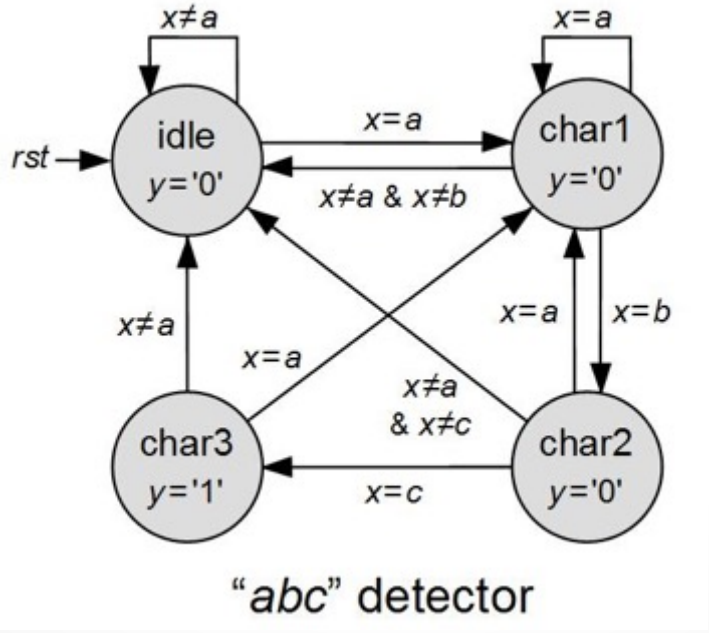
Bem forte
(vários estados, muitas interconexões, mais do que uma entrada, possivelmente envolvendo tempo)

FSM

- Representações para FSMs:
  - Do ponto de vista das especificações
    - Diagrama de transição de estados
  - Do ponto de vista do hardware
    - Seção combinacional + seção sequencial

FSM

- Exemplo: detector de padrão "abc"
  - Diagrama de transição → Estados, transições, saídas



FSM

- Representações para FSMs:
  - Do ponto de vista das especificações
    - Diagrama de transição de estados
  - Do ponto de vista do hardware
    - Seção combinacional + seção sequencial

FSM

- Do ponto de vista do hardware
  - Seção combinacional + seção sequencial

Diagrama de transição de estados

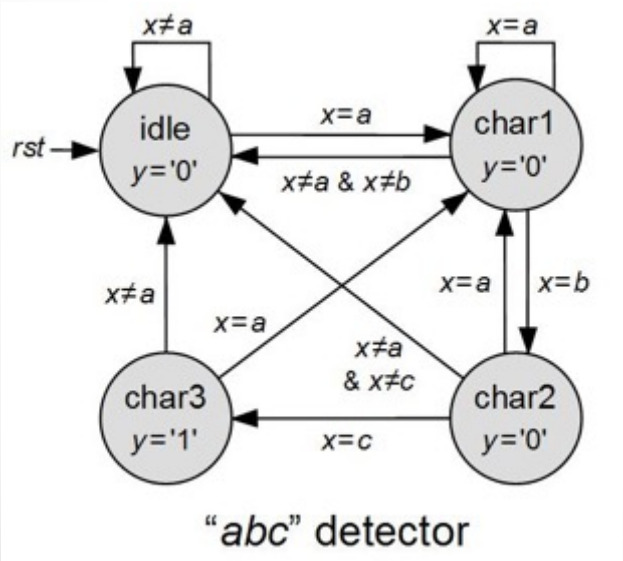
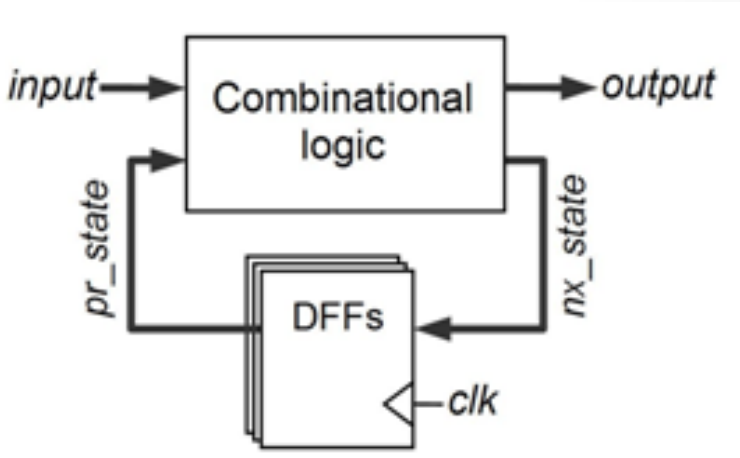
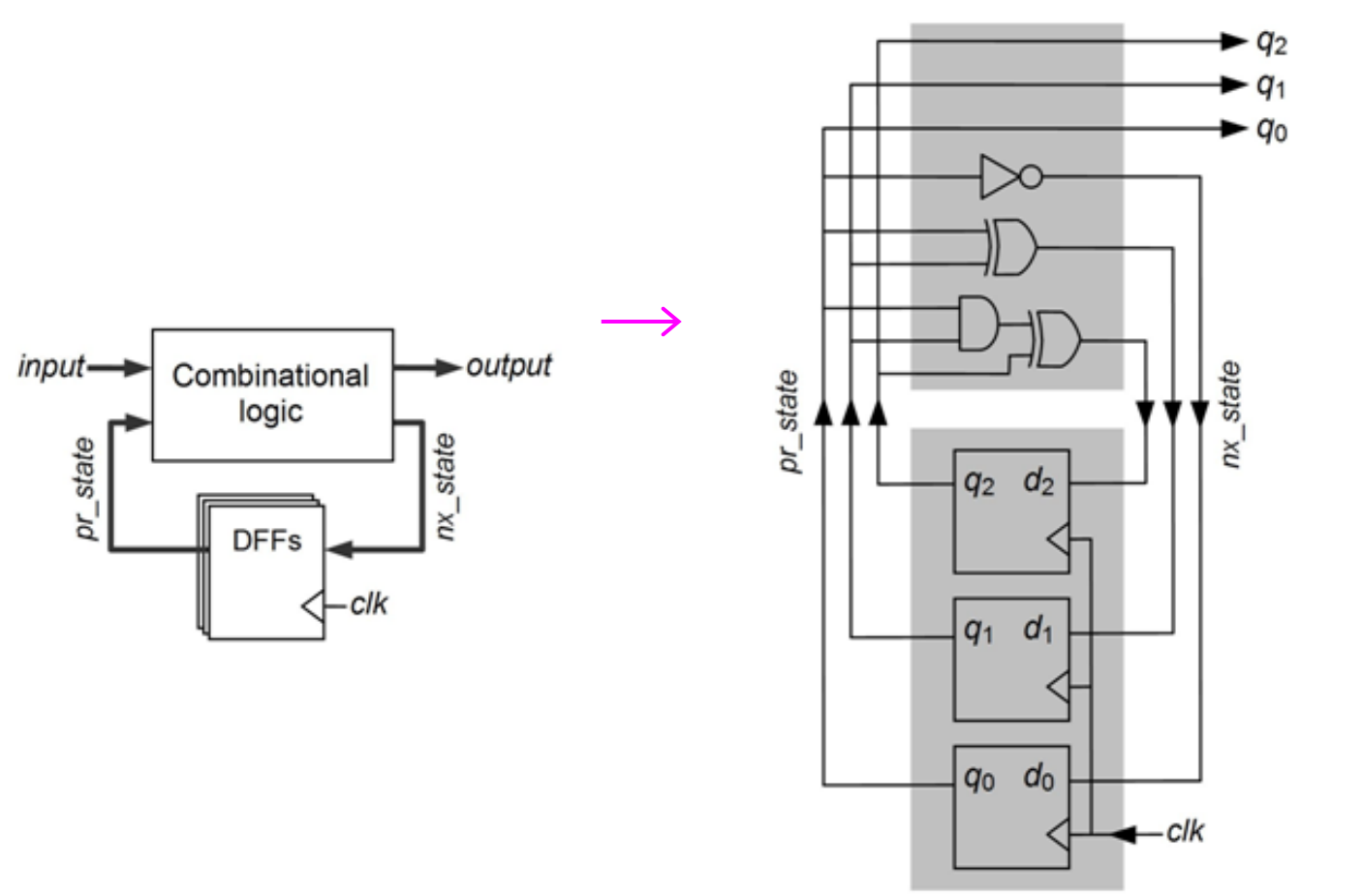


Diagrama do hardware



FSM

- Exemplo


input
Combinational logic
output
pr_state
nx_state
DFFs
clk
q2
q1
q0
pr_state
nx_state
q2 d2
q1 d1
q0 d0
clk

Tipos de Máquinas

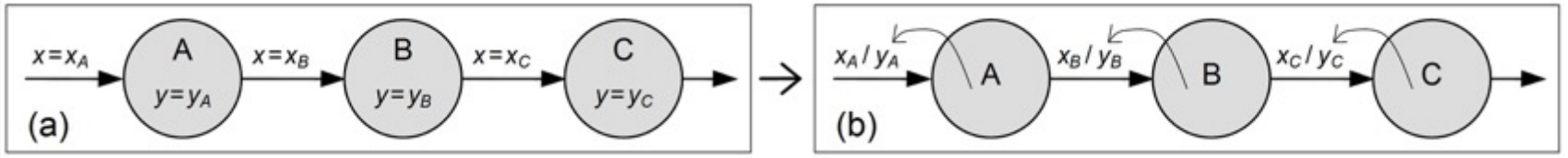
Tipos de Máquinas

- Existem dois tipos de FSMs
  - **Moore** → Saída só depende do estado em que a máquina está
  - **Mealy** → Saída depende do estado da máquina e também da entrada

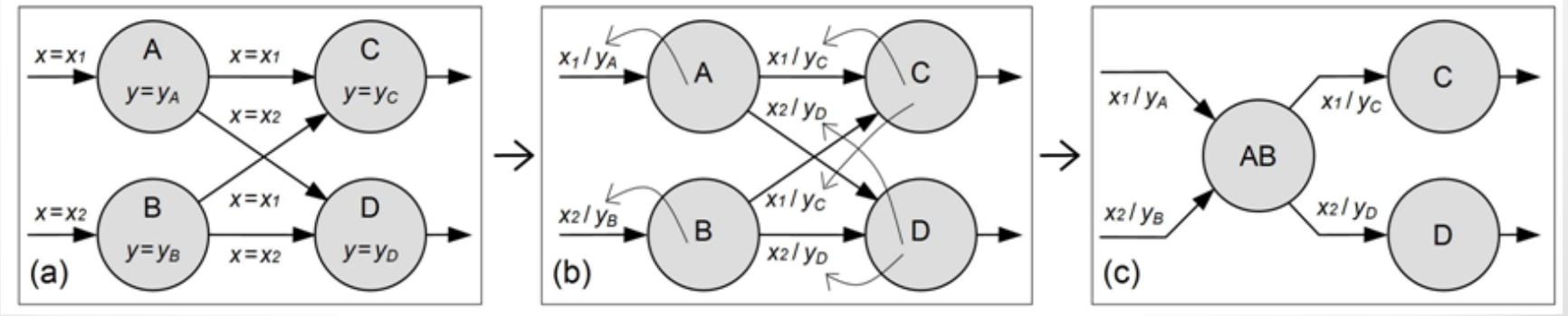
- Logo:
  - **Moore** é sempre síncrona; **Mealy** pode ser assíncrona.
  - Usa-se **Mealy** quando quer-se efeito imediato da entrada sobre a saída

Tipos de Máquinas

- Princípio de conversão de Moore para Mealy



- Exemplo



Tipos de Máquinas

- Dois estados são equivalentes (como A e B em (b) acima) quando:
  - As condições de transição são iguais
  - Os valores na saída são também iguais
- Número de estados da Mealy nunca será maior do que na Moore

FSM

- Para concluir:
  - Os benefícios da técnica FSM são:
    - Não precisa saber a priori o tipo de circuito a ser utilizado
    - É uma técnica sistemática (um método, sempre igual)
    - A solução será sempre, no mínimo, próxima de ótima

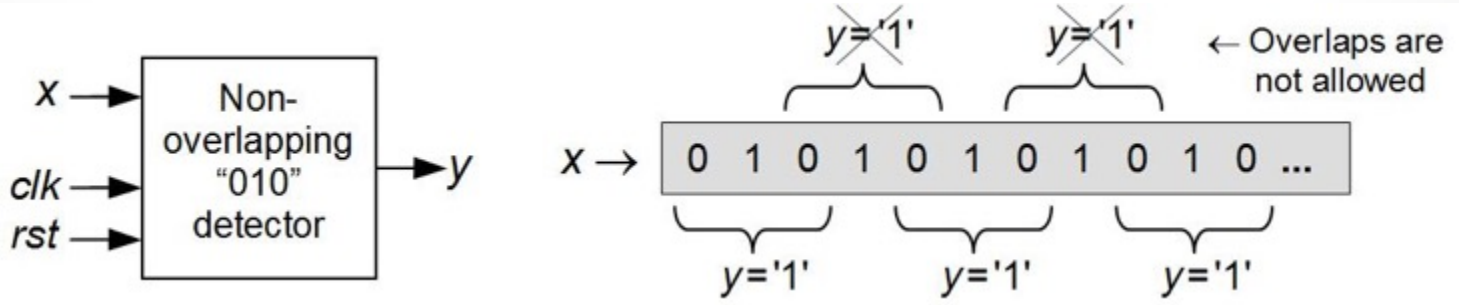
Procedimento de Projeto

Procedimento de Projeto

- Quatro passos:
  - **Passo 1:** Desenhe o diagrama de estados.
  - **Passo 2:** Escreva as tabelas-verdade para **nx_state** e **output**. Rearranje então as tabelas substituindo nomes por valores binários.
  - **Passo 3:** Obtenha as equações booleanas para **nx_state** e **output**.
  - **Passo 4:** Desenhe o circuito, colocando todos os flip-flops (somente DFFs) na seção inferior e a lógica combinacional (para as equações derivadas acima) na seção superior.

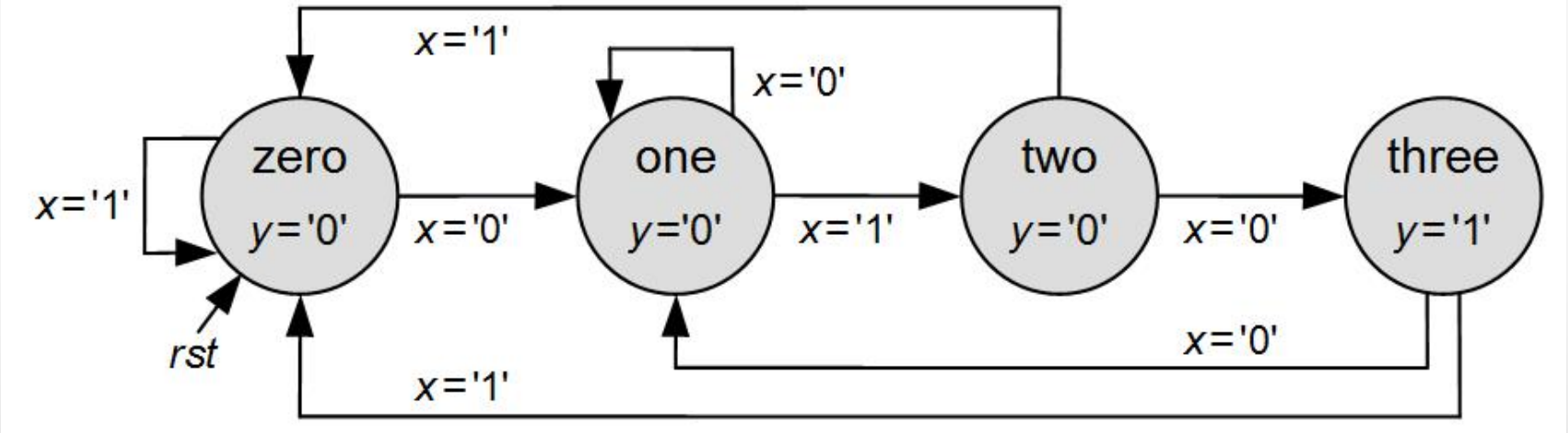
Procedimento de Projeto

- Exemplo
  - Detector da sequência “010” sem overlap



Procedimento de Projeto

- Exemplo
  - **Passo 1:** Diagrama de Transição



Procedimento de Projeto

- Exemplo
  - **Passo 2:** Tabelas Verdade e Codificação

Truth table for next state

pr_state	*x*	*nx_state*
zero	0	*one*
	1	*zero*
one	0	*one*
	1	*two*
two	0	*three*
	1	*zero*
three	0	*one*
	1	*zero*

→

Truth table for next state

pr_state q_1 q_0	*x*	*nx_state* d_1 d_0
0 0	0	0 1
	1	0 0
0 1	0	0 1
	1	1 0
1 0	0	1 1
	1	0 0
1 1	0	0 1
	1	0 0

Truth table for output

pr_state	*y*
zero	0
one	0
two	0
three	1

→

Truth table for output

pr_state q_1 q_0	*y*
0 0	0
0 1	0
1 0	0
1 1	1

Procedimento de Projeto

- Exemplo
  - **Passo 3:** Mapas-K e Equações

Karnaugh for d_1:

x \ $q_1 q_0$	00	01	11	10
0	0	0	0	1
1	0	1	0	0

$d_1 = q_1' \cdot q_0 \cdot x + q_1 \cdot q_0' \cdot x'$

Karnaugh for d_0:

x \ $q_1 q_0$	00	01	11	10
0	1	1	1	1
1	0	0	0	0

$d_0 = x'$

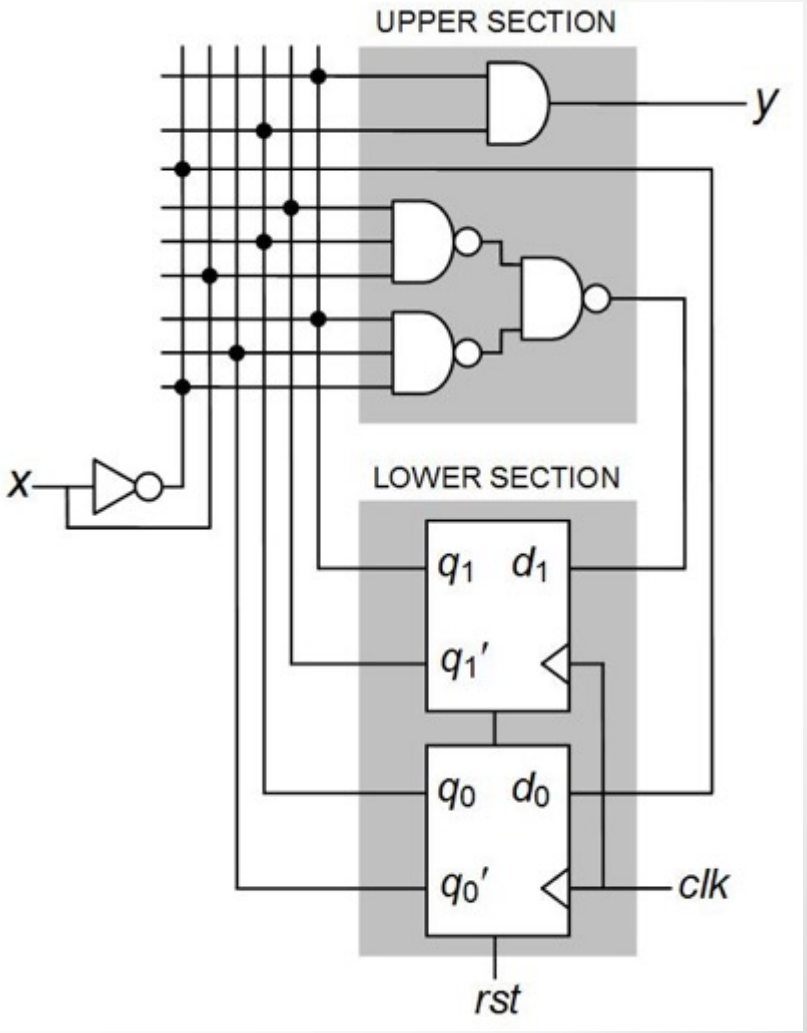
Karnaugh for y:

q_0 \ q_1	0	1
0	0	0
1	0	1

$y = q_1 \cdot q_0$

Procedimento de Projeto

- Exemplo
  - **Passo 4:** Circuito



Procedimento de Projeto

- Lembrando:
  - 4 passos

1. Diagrama de Transição (+importante!)
2. Tabelas Verdade e Codificação
3. Equações
4. Circuito

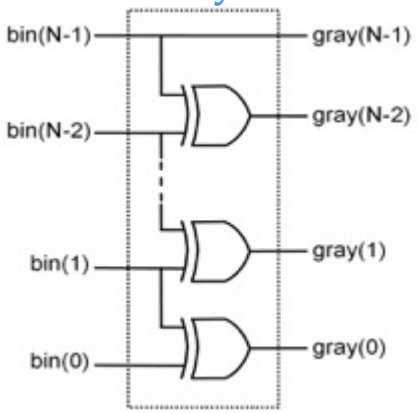
Estilos de Codificação dos Estados

Codificação

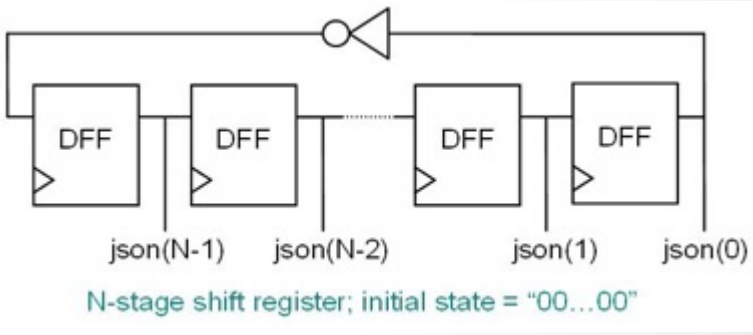
- Estilos de Codificação
  - Sequencial
  - Gray
  - Johnson
  - One-hot
  - Definido pelo Usuário

Exemplos: Contadores Gray/Johnson/One-hot

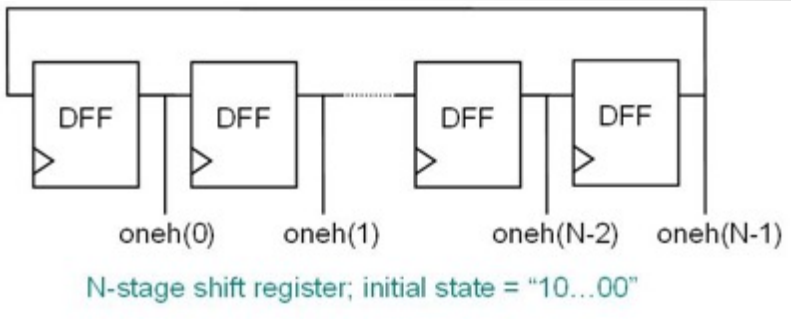
Gray



Johnson



One-hot



Codificação

- Estilos de Codificação
  - Quantos estados podem ser codificados com N=4 FF usando:
    - Código Sequencial
    - Código Gray
    - Código Johnson
    - Código One-hot

Codificação

- Estilos de Codificação

FSM encodings (for 4-bit systems)					
Sequential		Gray		Johnson	One-hot
0000	1000	0000	1100	0000	0001
0001	1001	0001	1101	1000	0010
0010	1010	0011	1111	1100	0100
0011	1011	0010	1110	1110	1000
0100	1100	0110	1010	1111	---
0101	1101	0111	1011	0111	---
0110	1110	0101	1001	0011	---
0111	1111	0100	1000	0001	---
Total=16 states		Total=16 states		Total=8 states	Total=4 states

Codificação

- Estilos de Codificação
  - Código sequencial ou Gray
    - Número de flip-flops é mínimo
    - Lógica combinacional pode ser um pouco maior e mais lenta
  - Código one-hot
    - Maior número de flip-flops
    - Lógica combinacional pode ser um pouco menor e mais rápida
  - Código Johnson
    - Intermediário aos acima

Problema de Sub & Sobre Especificação

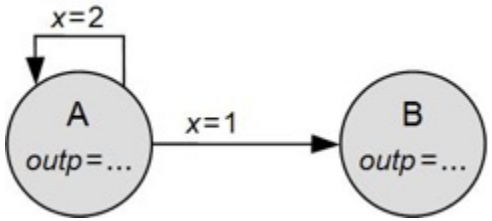
Sub & Sobre Especificação

- As condições de transição devem ser realmente complementares. Cuidado ao projetar sua FSM para não ter condições:
  - **Sub-especificadas:** com condições não cobertas
  - **Sobre-especificadas**: com condições cobertas mais do que uma vez

Sub & Sobre Especificação

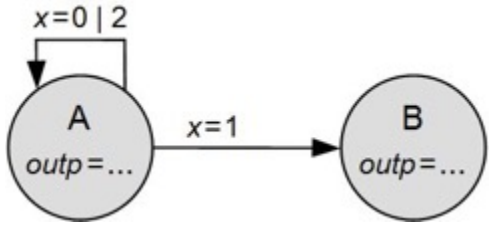
- Exemplo

Assumido:
$x = \{0, 1, 2\}$

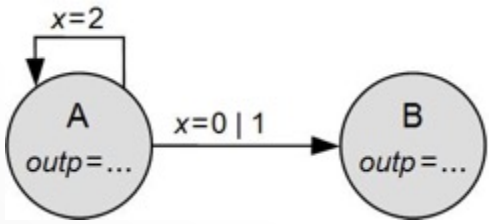


Consequências:
- Inferência de latches
- Next state é acidental

Corrigido
(caso que faltava foi assinalado à transição AA)



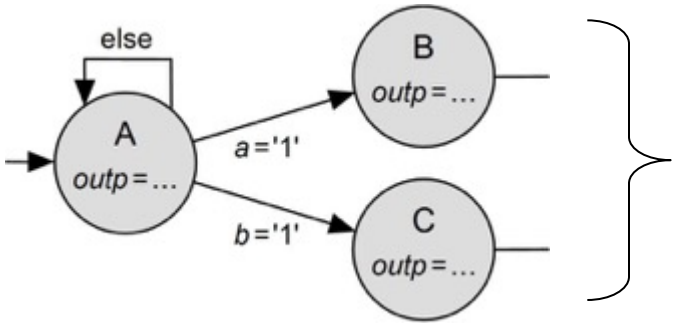
Corrigido
(caso que faltava foi assinalado à transição AB)



Sub & Sobre Especificação

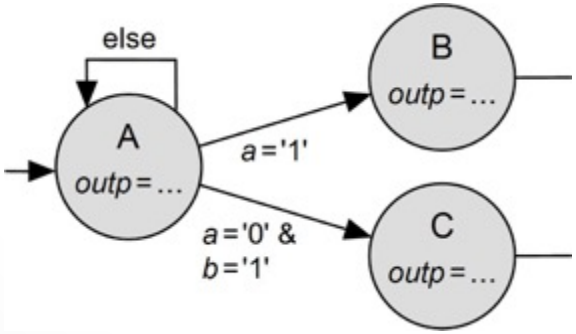
- Exemplo

Assumido:
a e *b* são sinais de um bit



Consequência:
- Next state é acidental

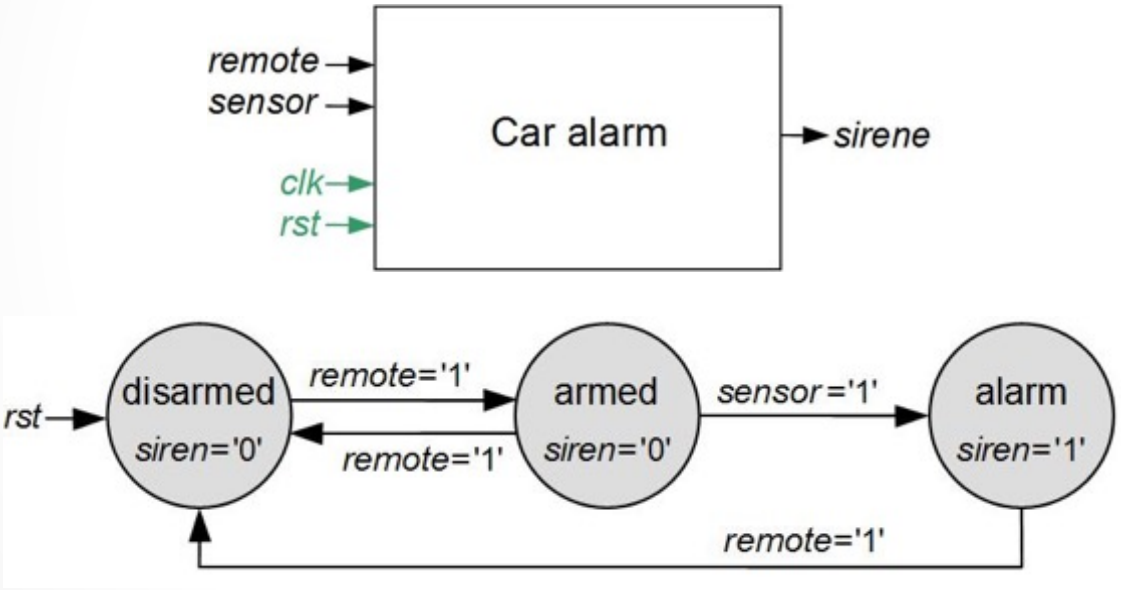
Corrigido:
a tem prioridade sobre *b*



Problema de "State Bypass"

State Bypass

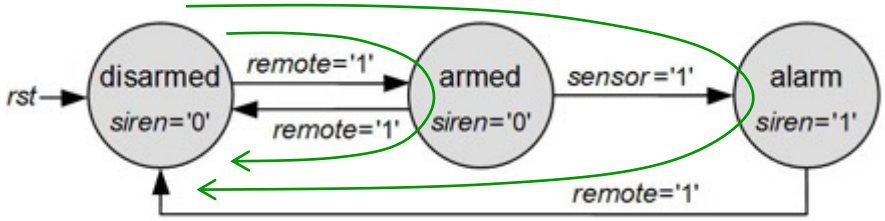
- Exemplo: Alarme de Carro



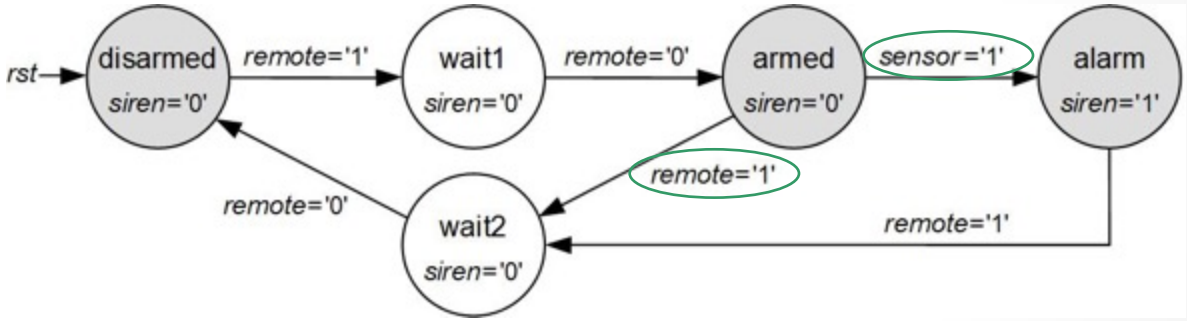
- O que ocorre se *remote* = '1' dura diversos ciclos de clock?

State Bypass

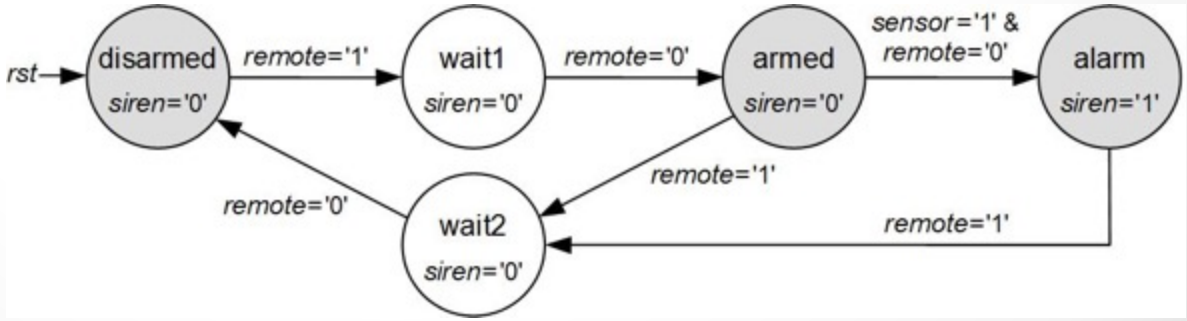
State-bypass:



Corrigido:
MESMO?



Final:



State Bypass

- O problema de State Bypass é uma análise de usabilidade e vai além de um problema "meramente" técnico

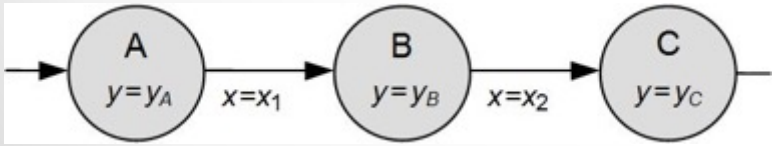
- O papel dx engenheirx é criar soluções não apenas que atendem o mínimo critério técnico mas também de usabilidade, orçamento, manutenabilidade, etc.

Máquinas Temporizadas

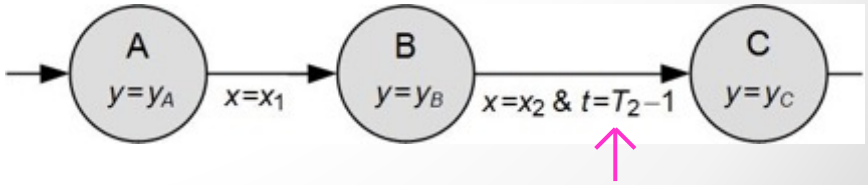
Máquinas Temporizadas

- Lembrar dos tipos de transições:
  - Condicional
  - Temporizada (caso especial da condicional)
  - Condicional-temporizada
  - Incondicional

Máquina regular (básica):



Máquina temporizada:



Máquinas Temporizadas

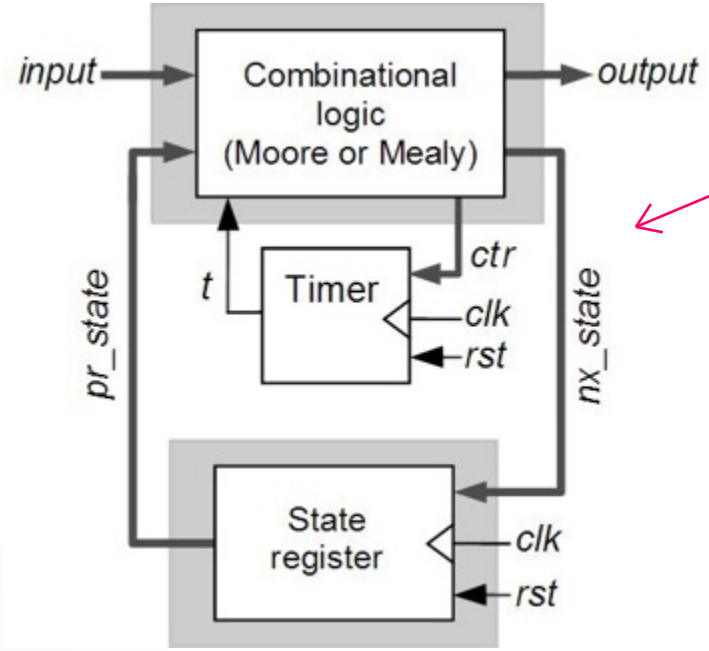
- Procedimento de projeto para máquinas temporizadas:
  - Utilize um timer para controlar o tempo
  - O timer é apenas um Contador simples
  - O timer é externo à máquina

Máquinas Temporizadas

- Procedimento de projeto para máquinas temporizadas:
  - Remova o timer da FSM e use sua saída como uma entrada qualquer da FSM
  - O timer e a FSM são independents mas o timer irá ser controlado pelos valores de **pr_state** e **nx_state** da máquina

Máquinas Temporizadas

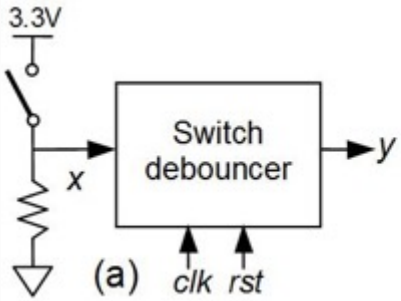
Arquitetura resultante:

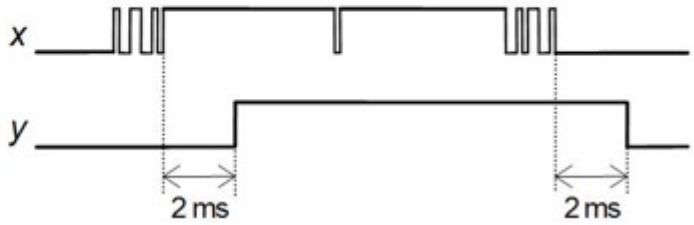


IMPORTANTE:
A FSM é quem controla o timer

Máquinas Temporizadas

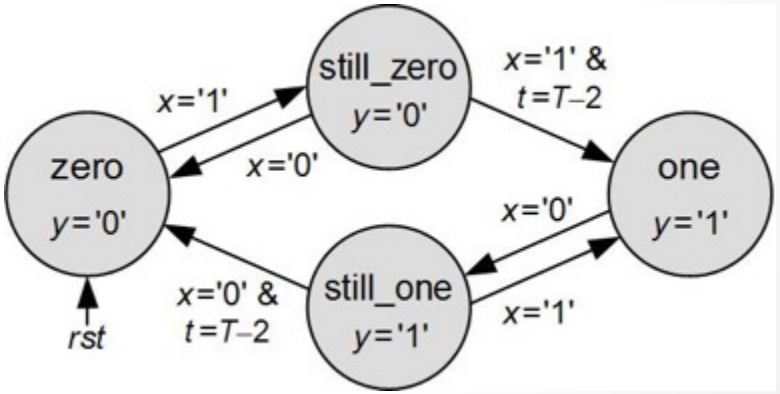
Exemplo: Debounceador de chave





Estratégia de controle do timer:

1) Pare o timer quando ele atingir o valor monitorado.
2) Zere o timer quando a FSM mudar de estado.

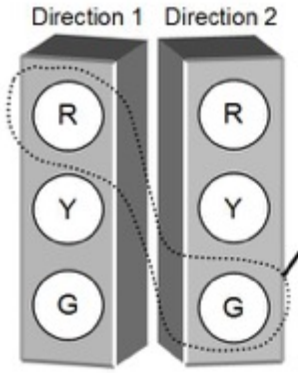


Por que *T*-2?

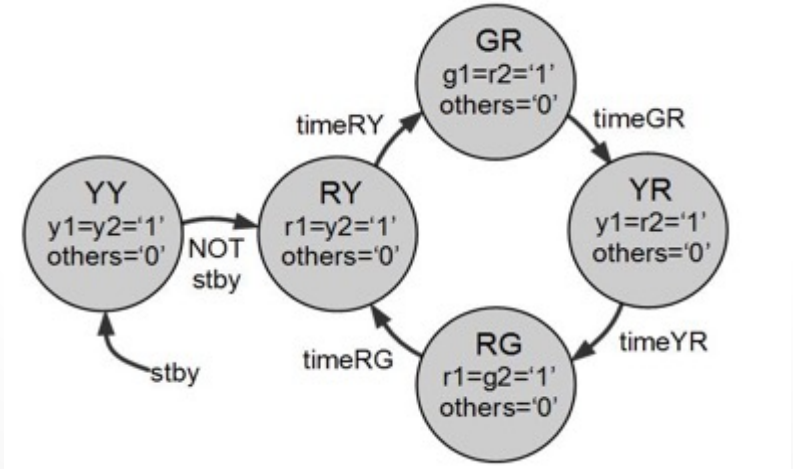
Porque um ciclo de clock foi gasto na transição anterior.

Máquinas Temporizadas

Outro exemplo: Controlador de semáforo



State	Operating modes		
	Regular	Test	Standby
	timer:	timer:	timer:
RG	timeRG (30s)	timeTEST (1s)	---
RY	timeRY (5s)	timeTEST (1s)	---
GR	timeGR (45s)	timeTEST (1s)	---
YR	timeYR (5s)	timeTEST (1s)	---
YY	---	---	Indefinite



Conclusão

Conclusão

- FSM é uma técnica sistemática, com um passo a passo bem definido
- Não serve para todos os projetos – avaliar de antemão o uso
- Gera circuitos ótimos (ou próximo disso)
- FSM Regulares e FSM Temporizadas

O Fim.

Perguntas? Não?

Então palmas para o professor.